Arquitectura de Integración Orientada a Servicios

Mario Bolo
bolo@ar.ibm.com

IBM Certified Senior IT Architect
IBM Software Group, SSA

Resumen

En el mundo de los negocios de hoy, las empresas necesitan tener una gran flexibilidad para poder adaptarse ágilmente a lo que un entorno muy exigente les demanda. Las arquitecturas IT tradicionales no han podido dar una respuesta adecuada a esta necesidad debido fundamentalmente a que en ellas las aplicaciones están diseñadas generalmente como "silos verticales", pensadas para un propósito específico y limitado, de modo que su integración resulta muy trabajosa. Esto dificulta a su vez la rápida adaptación de los procesos para poder aprovechar nuevas oportunidades de negocio o responder a amenazas externas. Hoy en día ha surgido una nueva forma de concebir los sistemas de información, denominada Arquitectura Orientada a Servicios (SOA), que parece brindar la mejor respuesta de la que se dispone hasta ahora para dotar a las organizaciones de la flexibilidad requerida. En una SOA, la funcionalidad aplicativa se brinda a través de componentes denominados servicios, que presentan interfaces estándar bien definidas y que representan funciones de negocio. Los servicios se pueden combinar en lo que se denomina coreografía de servicios, permitiendo implementar de ese modo procesos de negocio sumamente ágiles y flexibles. Naturalmente la implementación de una SOA requiere nuevas herramientas de software. El presente artículo describe las herramientas de software de IBM que pueden ayudar a implementar una infraestructura SOA.

Palabras clave:
SOA (Service-Oriented Architecture), Servicios, Arquitectura, Arquitectura orientada a servicios, Procesos de negocio, Web Services, Coreografía, Integración, EAI (Enterprise application integration).

1. Introducción

Los sistemas de información han recorrido durante los últimos 20 años un camino que lleva desde los primitivos sistemas monolíticos, pasando por sistemas cliente/servidor, difíciles de administrar, hasta las arquitecturas basadas en servicios, en las que la funcionalidad de las aplicaciones se implementa en la forma de componentes reutilizables e invocables mediante interfaces estándar, que pueden combinarse para crear funciones progresivamente más complejas. Hoy en día toda la industria, tanto los proveedores de tecnología como los consultores y los usuarios, están de acuerdo en que este último enfoque permite mejorar la calidad de las aplicaciones y dar una mejor respuesta a las necesidades de los negocios. En los próximos párrafos intentaremos explicar por qué esto es así.

Una de las principales necesidades que enfrentan hoy los negocios es la de integrar sus procesos en forma "transversal" a través de los diferentes aplicativos, y ésta es precisamente una necesidad a la cual las arquitecturas IT anteriores casi nunca pudieron dar una respuesta completamente satisfactoria. Tradicionalmente la integración de los procesos se llevó a cabo, en algunos casos manualmente, obligando a los usuarios a conectarse a diferentes aplicaciones (y en muchos de esos casos a utilizar un *sign-on* diferente cada vez), o a llevar papeles de un sector de la empresa a otro, con las consiguientes inexactitudes e ineficiencias; en otros casos la lógica de integración de los procesos estaba "escondida" dentro del código aplicativo, haciendo difícil el poder modificar ágilmente los procesos o el crear procesos nuevos reutilizando los componentes existentes.

En el mundo de los negocios de hoy, las empresas necesitan tener una enorme flexibilidad para poder adaptarse ágilmente a lo que un entorno muy exigente les demanda. Esto es lo que IBM ha denominado *e-Business On Demand* y se refiere a empresas cuyos procesos de negocio –integrados de punta a punta dentro de la organización y también con otras organizaciones, con los clientes y con los proveedores– pueden adaptarse con flexibilidad y rapidez a cualquier demanda de los clientes, oportunidad del mercado o amenaza externa. Según esta visión, las empresas que logren convertirse en negocios *on-demand* ahorrarán costos, llegarán a un mercado más amplio, interactuarán más efectivamente interna y externamente, cubrirán mejor las demandas de sus clientes e incrementarán en definitiva sus beneficios.

Como ya hemos comentado antes, la dificultad de las arquitecturas tradicionales para integrarse en los procesos de negocio reside en que,

en ellas, las aplicaciones no están diseñadas para ser integradas con otras. En esas arquitecturas las aplicaciones se diseñan generalmente como "silos verticales", cada una de ellas pensada para un propósito específico y limitado. En muchos casos se trata de aplicaciones construidas o adquiridas en diferentes momentos históricos, por diferentes equipos de gente y en forma independiente, de modo que es natural que su integración resulte dificultosa.

Ese estado de cosas es obviamente indeseable ya que la infraestructura tecnológica no debería inhibir sino, por el contrario, facilitar la integración de los procesos de negocio, permitiendo además que dichos procesos reutilicen las aplicaciones existentes e incorporen también las nuevas aplicaciones que se desarrollen en el futuro, aun sobre diferentes plataformas. Lograr todo esto requiere que la infraestructura tecnológica se estructure de acuerdo con los principios de una *arquitectura tecnológica*, que brinde un marco que permita ensamblar con facilidad componentes y servicios para crear nuevas soluciones en cuanto el negocio las demande. El propósito principal de este documento es precisamente el de describir la arquitectura tecnológica de integración basada en servicios definida por IBM.

La Arquitectura Orientada a Servicios (SOA)

En el apartado anterior comentamos que hoy existe un amplio consenso en que la mejor manera de implementar la visión e-Business On Demand a la que nos referimos antes es mediante una *Arquitectura Orientada a Servicios (SOA)*. Pero, ¿qué es exactamente una SOA?

SOA es un modelo de componentes que interrelaciona las diferentes unidades funcionales de una aplicación, llamadas *servicios*, a través de interfaces bien definidas entre dichos servicios. Las interfaces se definen de una manera neutral, independiente de la plataforma de hardware, sistema operativo, o lenguaje de programación en el que el servicio se implementa. Esto permite que los servicios, construidos sobre una gran variedad de tecnologías, puedan interactuar unos con otros de una manera uniforme y universal.

Esta característica de disponer de una definición neutral de las interfaces, no fuertemente ligada a ninguna implementación particular, se conoce como *acoplamiento débil* entre servicios. Los beneficios de un sistema débilmente acoplado consisten en su agilidad y capacidad para sobrevivir a cambios evolutivos en la estructura e implementación de

cada uno de los servicios individuales que constituyen la aplicación en su totalidad. El acoplamiento fuerte, por otra parte, significa que los diferentes componentes de una aplicación están íntimamente relacionados en funcionalidad y forma, lo cual hace a estas aplicaciones muy vulnerables a los cambios evolutivos, ya que cualquier modificación a uno de sus componentes termina afectando a los demás. [1]

La necesidad de contar con sistemas débilmente acoplados surge de la necesidad ya mencionada de dotar a las aplicaciones de mayor agilidad, lo cual se basa a su vez en la necesidad del negocio de adaptarse rápidamente a su ambiente cambiante. Podría decirse que, en última instancia, el propósito de una SOA es desvincular las aplicaciones de las implementaciones de los componentes que dichos procesos utilizan. A esto se le llama "separación de las incumbencias" *("separation of concerns",* en inglés). La gran ventaja de esta separación es que permite cambiar la implementación de los componentes sin afectar las aplicaciones y, viceversa, modificar las aplicaciones reutilizando los mismos componentes. Es evidente que este modelo puede darle a los negocios la flexibilidad que los sistemas tradicionales no podían brindar.

Muchas de las ideas de SOA no son totalmente nuevas. Sin embargo una característica que diferencia a SOA de otras arquitecturas similares del pasado (tales como CORBA) es la importancia que ha adquirido en SOA un avance relativamente reciente, que aprovecha la amplia aceptación del XML como lenguaje común de descripción de datos y que consiste en utilizar para las interfaces un conjunto de especificaciones basadas en XML. Estas especificaciones son llamadas *Web Services*, y han sido adoptadas oficialmente por el comité que fija los estándares de la web, el W3C. [2]

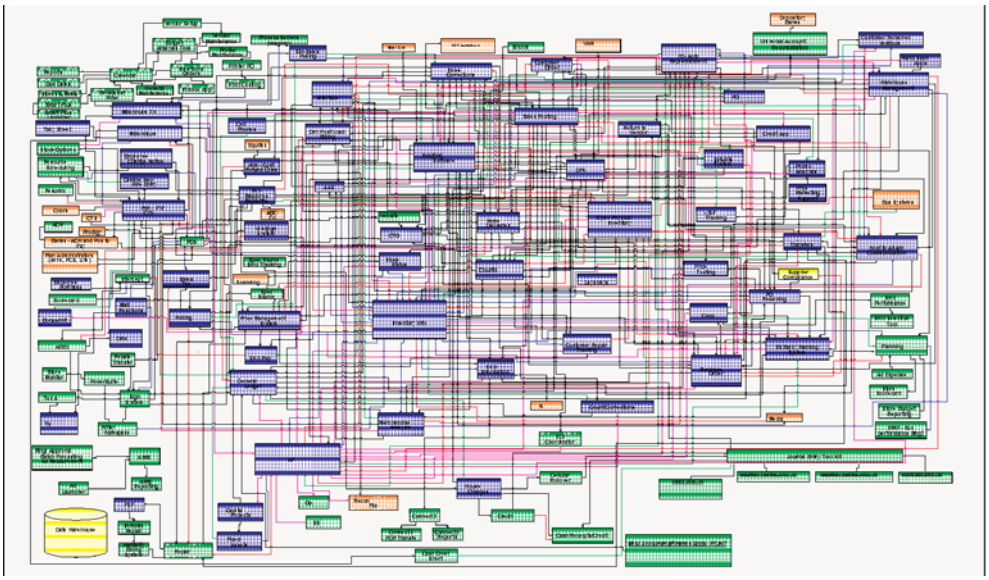
No deben confundirse sin embargo los planos: SOA es una arquitectura, una forma de concebir y diseñar sistemas mientras que los Web Services son una forma de implementar los servicios en una SOA, y no la única. Un *middleware* orientado a mensajería, tal como el *WebSphere MQ* puede proveer una forma perfectamente viable de implementar SOA. Por otra parte, un modelo de arquitectura necesita mucho más que una descripción de los servicios. Necesita definir, por ejemplo, cómo la aplicación lleva a cabo su flujo de trabajo *(workflow)* entre los servicios. Más aún, un modelo de arquitectura debería poder definir el punto de transformación entre las operaciones del negocio y las operaciones del software utilizado en el negocio. De ese modo, una SOA debería ser capaz de relacionar los procesos comerciales de un negocio con sus procesos técnicos. Por ejemplo, el pago a un proveedor es un proceso de

negocio mientras que actualizar la base de datos de partes para incluir la recepción de una nueva remesa es un proceso técnico. Una SOA debería poder relacionar ambos procesos en un mismo flujo de trabajo; de ese modo, vemos que el concepto de *workflow* juega un rol muy significativo en el diseño de una SOA.

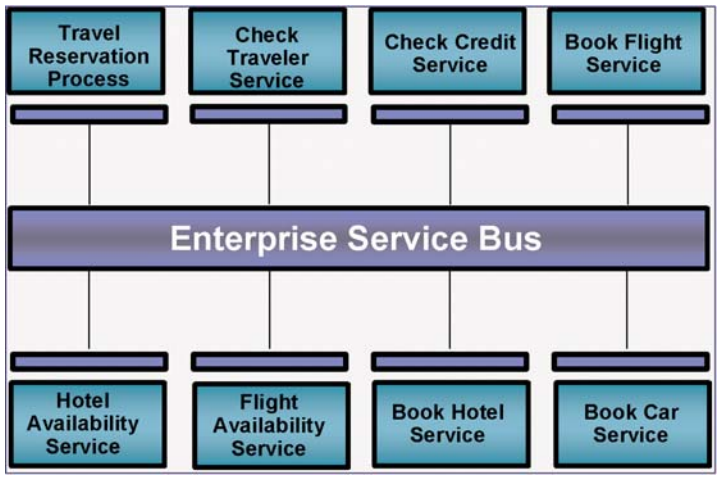
Yendo aún un paso más allá, un *workflow* de un negocio dinámico puede incluir operaciones no sólo internas, entre los departamentos de la propia empresa, sino también con asociados de negocio externos, sobre los que la empresa no tiene ningún control. Esto hace necesario en una SOA definir políticas operativas y acuerdos de niveles de servicios para definir la forma en que las empresas van a interactuar. Finalmente, todo esto tiene que operar en un ambiente confiable y seguro para que los procesos puedan llevarse a cabo en los términos acordados. Por lo tanto, la seguridad y la confiabilidad deberían jugar un rol significativo en SOA.

En una SOA los diferentes servicios habitualmente no interactúan en forma directa unos con otros sino que lo hacen utilizando la mediación de un *Enterprise Service Bus (ESB)*.

Lo que se desea evitar es una situación tal como la que muestra la siguiente figura, que es el mapa de las interfaces entre aplicativos tomado de un caso real (una empresa de productos electrónicos de consumo masivo):

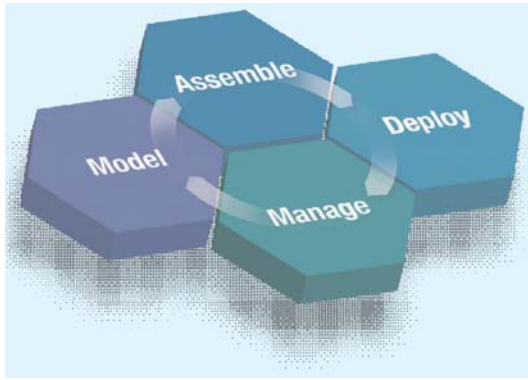


Un ESB es un *backbone* de integración, al cual se conectan los diferentes servicios y a través del cual fluyen los mensajes que permiten que aquellos interactúen, de modo que la arquitectura de integración de los servicios pasa a convertirse en algo como esto:



Un ESB no es simplemente un "cable" que conecta los diferentes servicios; un ESB es por el contrario un elemento que puede rutear inteligentemente cada requerimiento al componente que lo pueda brindar, en base al tipo de servicio requerido o inclusive a los datos del requerimiento. También posee la capacidad de reformatear los datos para adaptarlos a los diferentes aplicativos participantes y provee además facilidades de manejo de eventos. Vamos a ver este tema en mayor detalle más adelante. [3]

Ahora bien, la implementación de facilidades tales como las que acabamos de describir requiere un cierto número de capacidades o habilidades, que están íntimamente relacionadas con el ciclo de vida de los procesos de negocio, tal como se muestra en el gráfico siguiente:



Como muestra el gráfico, las capacidades necesarias para implementar SOA son:

- Modelar los procesos de negocio: el analista de procesos o especialista en métodos y procedimientos aplica su conocimiento del negocio para crear gráficamente un modelo del proceso y simular en su estación de trabajo los resultados de su ejecución (tiempos, costos, ingresos, recursos).
- Ensamblar los componentes necesarios: lo cual implica completar el proceso modelado en el paso anterior con los elementos técnicos necesarios (componentes J2EE, estructuras de datos, mensajes, etcétera) que posibiliten que aquel pueda efectivamente ejecutarse.
- Poner en marcha *(deployment):* es decir, poner a ejecutar el proceso ensamblado utilizando la infraestructura de software y hardware que sea necesaria, y que puede incluir elementos tales como: un motor de procesos, un ESB, etcétera.
- Administrar los procesos: o sea, monitorear su ejecución para poder corregir en tiempo real posibles desviaciones y situaciones de excepción que puedan estar provocando, por ejemplo, demoras indeseables, y para poder evaluar los resultados de la ejecución contra las metas de negocio definidas.

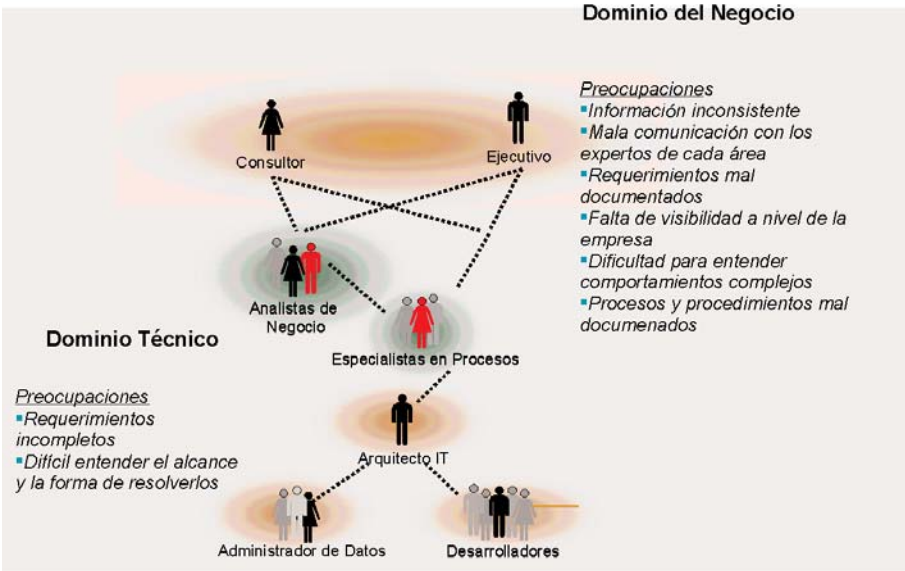
La evaluación de los resultados de los procesos puede llevar en muchos casos a la decisión de rediseñarlos para poder cumplir mejor con las metas estratégicas del negocio, lo cual nos conduce nuevamente al modelado, cerrando de ese modo el ciclo.

Disponer de una infraestructura de *middleware* que brinde todas estas capacidades simplifica enormemente la implementación de SOA, ya que permite a los participantes del proyecto concentrarse en las funciones específicas de negocios, tales como la lógica de los procesos, la implementación óptima de los componentes, etcétera, dejando en manos del *middleware* las capacidades de base.

Lo que IBM ha hecho es justamente definir una plataforma de *middleware* que brinda todas esas capacidades para que sus clientes y asociados de negocio puedan aprovecharlas en el desarrollo de sus nuevos aplicativos orientados a servicios. Vamos a describir a continuación la forma como IBM implementa cada una de las capacidades mencionadas.

Modelado, ensamblado y monitoreo de los procesos

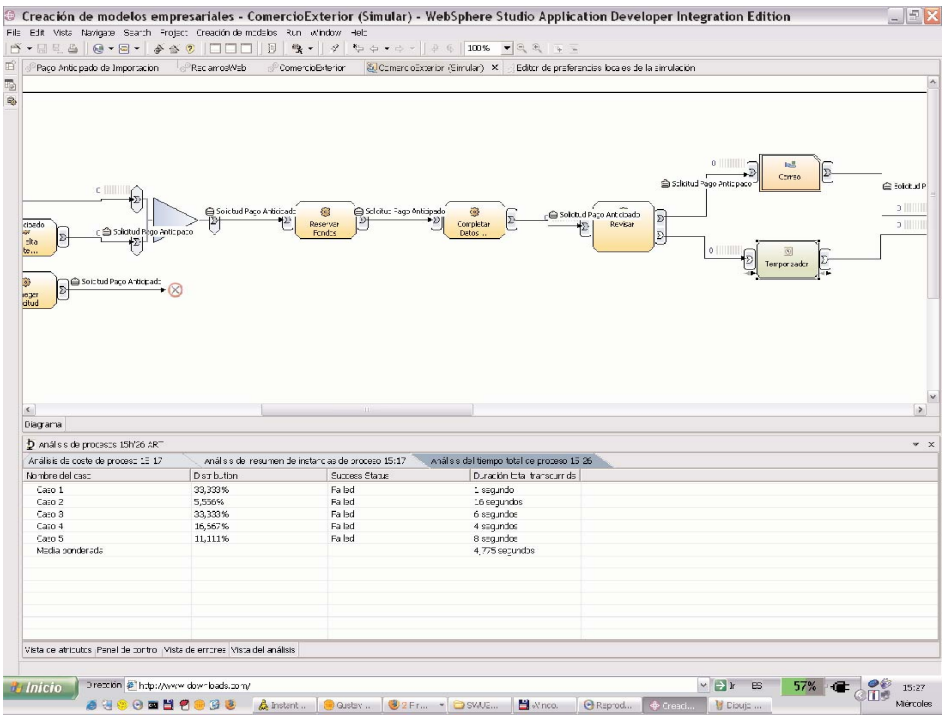
El modelado de los procesos es fundamental porque hace de "puente" entre la problemática del mundo de los negocios y la del mundo IT, tal como muestra la figura:



Utilizando una herramienta de modelado como *IBM WebSphere Business Modeler*, los analistas de negocio y especialistas en procesos pueden tomar como *input* los requerimientos definidos por los ejecutivos y los consultores en estrategia, para:

- Modelar en forma gráfica los procesos de negocios que resuelven esos requerimientos de la empresa.
- Rediseñar rápidamente los procesos existentes para adecuarlos a nuevos requerimientos impuestos por las situaciones cambiantes del mercado.
- Simular en la estación de desarrollo la ejecución de los procesos para evaluar sus tiempos, costos y resultados.

El gráfico siguiente muestra una captura de pantalla de WebSphere Business Modeler:

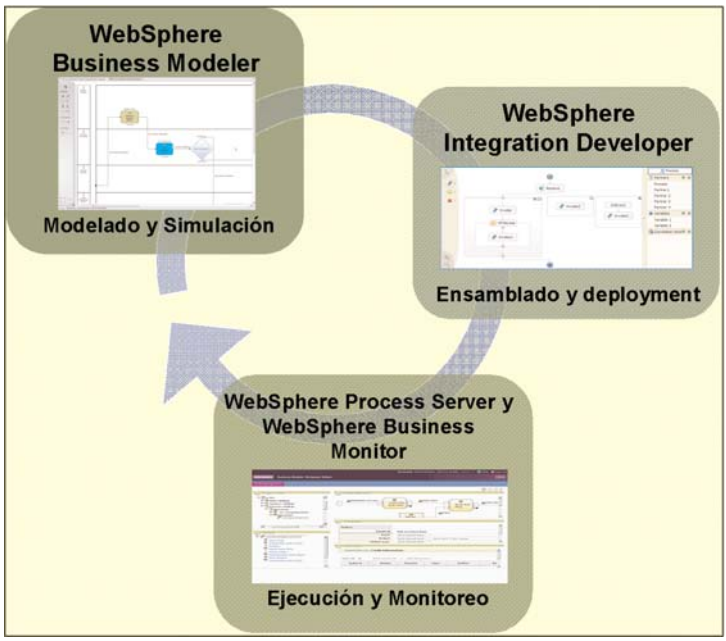


Los procesos modelados de este modo utilizando ***WebSphere Business Modeler*** pueden ser exportados en formato *BPEL (Business Process Execution Language)*. BPEL es un lenguaje para la descripción de procesos de negocio basado en XML y adoptado como estándar por la organización OASIS. Forma parte de lo que se suele llamar la segunda generación de Web Services.

El proceso en formato BPEL puede ser importado en una herramienta de desarrollo tal como ***IBM WebSphere Integration Developer*** para que los desarrolladores lo tomen y lo ensamblen con los restantes componentes tecnológicos necesarios para su puesta en marcha bajo el control de un motor de ejecución (***WebSphere Process Server***). De ese modo, el modelo del proceso, que expresa las necesidades de negocio de la empresa sirve además al propósito de comunicar mucho mejor dichas necesidades a los desarrolladores, ayudando así a alinear a IT con los negocios.

La utilización del modelado como parte del ciclo integral de los procesos maximiza los beneficios que pueden obtenerse de esa actividad. Sin

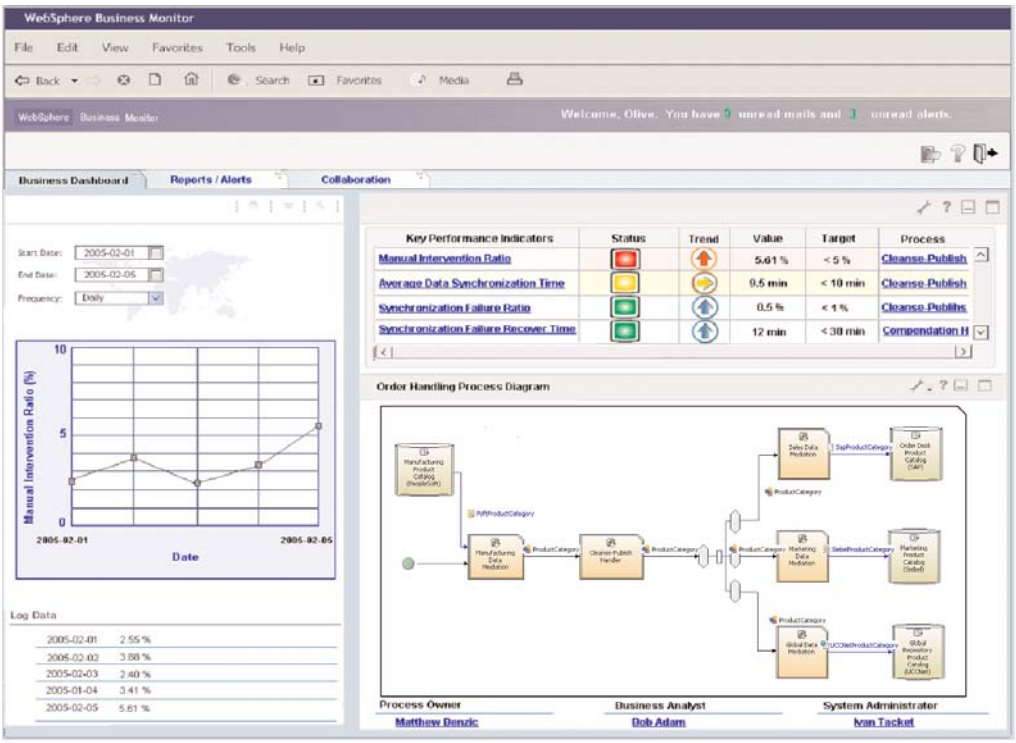
embargo no debe perderse de vista el hecho de que el modelado de los procesos tiene un valor en sí mismo. Muchas organizaciones aprovechan herramientas como *WebSphere Business Modeler* para diseñar, simular, documentar y publicar en la intranet sus procesos de negocio, aún cuando todavía no tengan planes inmediatos de ponerlos en ejecución bajo un servidor de procesos.



Volviendo al ciclo integral de los procesos de negocio, vemos en el gráfico anterior que, una vez que el proceso está ejecutándose bajo el control de WebSphere Process Server, puede ser monitoreado mediante el *WebSphere Business Monitor*. Este componente de monitoreo permite:

- Mostrar en tiempo real la operación de la empresa en tableros personalizados
- Enviar alertas para poder tomar una acción a tiempo ante situaciones anormales
- Rastrear activamente eventos de negocios a lo largo de su ejecución a través de la cadena de valor

La figura siguiente muestra un tablero de control (dashboard) ejemplo construido con el WebSphere Business Monitor:



WBI Monitor permite monitorear la performance de los procesos, mejorar el control sobre el negocio, asignar recursos más eficientemente y tomar acciones (tales como balancear la carga de trabajo, suspender un proceso, etcétera). WBI Monitor brinda también información estadística sobre procesos pasados, permitiendo tomar decisiones sobre la base de información tomada del mundo real. [4]

La ejecución de los procesos

Los elementos de una SOA: Arquitectura de Referencia IBM

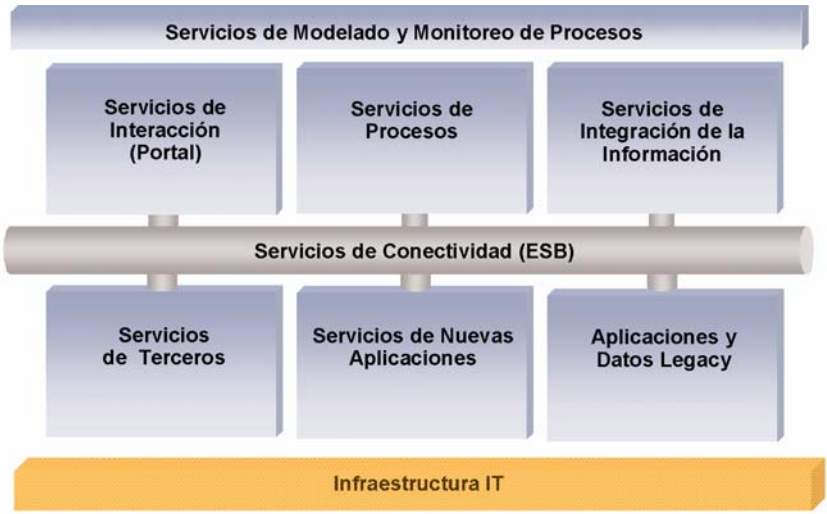
Los procesos de negocio en el contexto de una SOA se ejecutan bajo el control de un motor o servidor de procesos tal como el *WebSphere Process Server*, ya mencionado en la sección anterior de este documento. Existen además diversos componentes de *middleware* que facilitan la creación de nuevos servicios, la integración de los procesos con servicios nuevos o ya existentes y la interacción de los servicios y procesos con las personas.

En la visión propuesta por IBM estos componentes se clasifican del siguiente modo:

- Servicios de conectividad, que se implementan, como ya se ha dicho, mediante un Enterprise Service Bus (ESB).
- Servicios de negocio, que pueden ser dados por:
  - Nuevos componentes aplicativos, diseñados específicamente para una SOA y desarrollados por ejemplo en J2EE y ejecutarse bajo el servidor de aplicaciones *WebSphere Application Server*. IBM provee para el desarrollo de estos nuevos componentes los productos de la plataforma *IBM Rational* que cubren todo el ciclo de vida de las aplicaciones.
  - Componentes aplicativos preexistentes, que pueden integrarse a una SOA, como se ha dicho antes, a través de adaptadores tales como los *WebSphere Adapters* provistos por IBM, de los que hablaremos más adelante.
  - Servicios provistos por terceros. En las comunicaciones B2B se ha utilizado históricamente una gran variedad de protocolos (EDI, RosettaNet, FTP, Web Services, etcétera). Para aislar a los aplicativos y los procesos de la complejidad de manejar todos esos diferentes protocolos, y para manejar desde un lugar único cosas como la seguridad y los niveles de servicio, es conveniente contar con un *gateway*, que provea una interface de servicios a los aplicativos y procesos de la empresa y que se ocupe de manejar la comunicación con los diferentes proveedores de servicios externos. En la arquitectura SOA de IBM el componente que cubre esta función es el *WebSphere Partner Gateway*.

- Elementos de control de los servicios, que incluyen:
  - Servidor de procesos, el *WebSphere Process Server* ya mencionado y del cual hablaremos detalladamente más adelante.
  - Servidor de portal, que facilita la integración a nivel de la interface de usuarios de los diferentes aplicativos y procesos. En la arquitectura SOA de IBM esta funcionalidad la brinda el *WebSphere Portal Server*.
  - Servicios de integración de la información, que permiten visualizar fuentes de datos diversas como si fuesen una única base de datos. Esta funcionalidad la provee el *WebSphere Information Integrator*.

Todos estos elementos que acabamos de describir brevemente son los que forman la *Arquitectura SOA de Referencia de IBM*, que se esquematiza del siguiente modo:



A continuación apreciaremos en forma detallada algunos de los componentes de la Arquitectura de Referencia.

El Enterprise Service Bus (ESB)

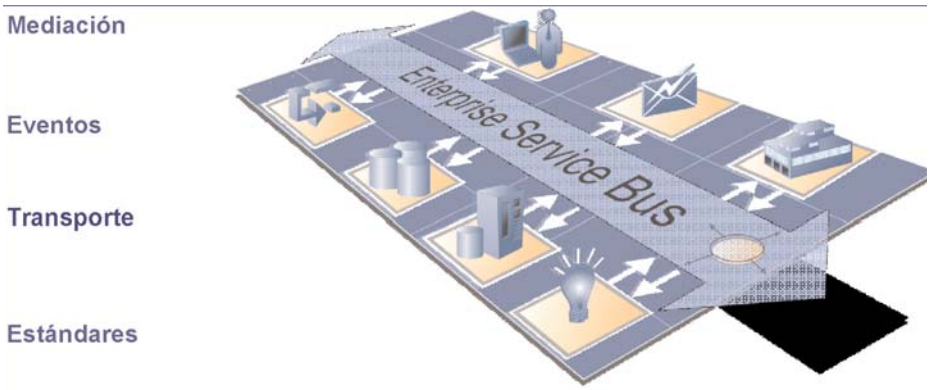
Hemos visto con anterioridad que, en una SOA, los servicios interactúan a través de un ESB. Un *ESB* no solo transporta mensajes entre los servicios sino que además provee una *mediación* entre ellos. El concepto de

mediación incluye:

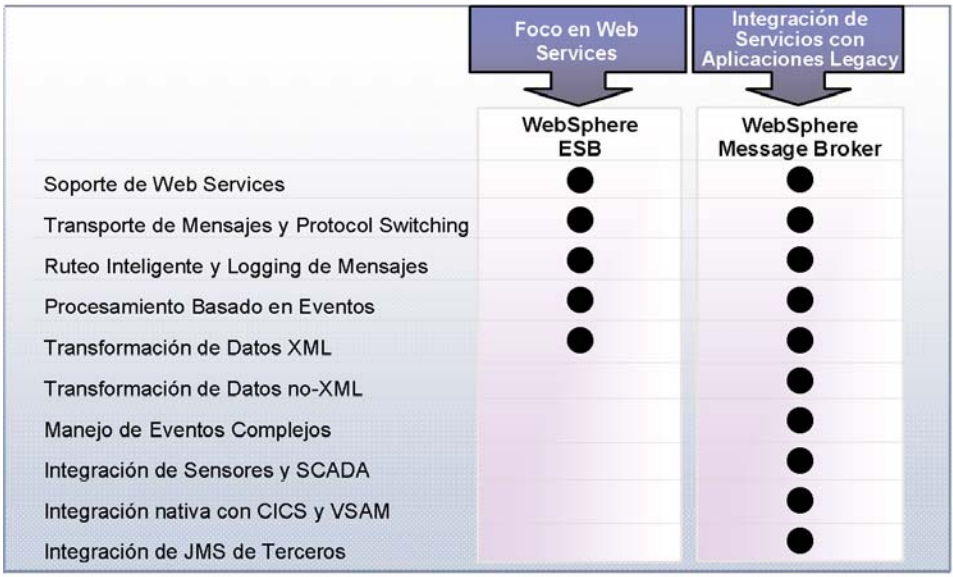
- *Ruteo*, que es la capacidad del ESB de derivar cada requerimiento de servicio al componente que deba procesarlo. Esto debe hacerlo el Bus inteligentemente, sobre la base del tipo de mensaje o de la información que el mensaje de requerimiento transporta.
- *Transformación*, que es la capacidad del ESB de modificar el formato de la información transportada por un mensaje para adecuarla al formato requerido por el proveedor del servicio.

El ESB soporta además el manejo de *eventos*. Esto significa que, cuando en una aplicación se produce un evento (por ejemplo: la actualización de un determinado dato), el ESB detecta ese evento y lo propaga a otras aplicaciones. Esta facilidad puede utilizarse por ejemplo cuando hay datos duplicados en varios sistemas, lo que origina el problema de mantener esos datos en permanente sincronismo para evitar inconsistencias en la información. Una forma de manejar este problema consiste justamente en que el ESB detecte el evento de la actualización de dicho dato para poder informar de ese evento a las restantes aplicaciones involucradas que podrán tomar así la acción que corresponda.

IBM provee dos productos para implementar un Enterprise Services



Bus. Uno de ellos es el *WebSphere Message Broker* (anteriormente conocido como MQ Integrator), que es el ESB avanzado, con soporte de múltiples protocolos de transporte y todo tipo de formatos de la información. El otro es un nuevo producto llamado *WebSphere ESB* y es un producto más sencillo y económico orientado exclusivamente a XML. El gráfico siguiente muestra las diferencias entre ambos productos:



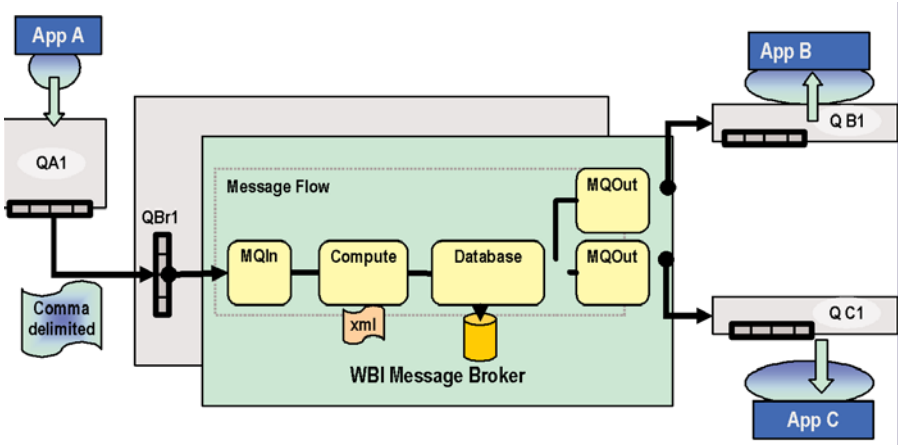
WebSphere Message Broker

WebSphere Message Broker provee un ESB avanzado que soporta todo tipo de protocolos de transporte (Web Services, JMS, MQ, FTP, SCADA, etcétera) y todo tipo de formatos de datos (XML, Cobol copybook, campos separados por comas, propietarios y otros). Tiene la capacidad de hacer *protocol switching* (por ejemplo, recibir un requerimiento de servicio sobre MQ y rutearlo al proveedor del servicio sobre HTTP) y todo tipo de transformaciones de formato (por ejemplo, de Cobol copybook a XML).

WebSphere Message Broker tiene un repositorio donde se almacenan los formatos de los diferentes mensajes. Estos formatos pueden ser creados, utilizando el editor de mensajes provisto por el propio producto o importándolos desde XML (DTDs o Schemas), copybooks Cobol o estructuras C / C++.

Un concepto central en WebSphere Message Broker es el de message-flow. Un message-flow consiste en un conjunto de nodos lógicos interconectados, donde cada nodo implementa una función específica de transformación o de ruteo. Los message-flows son, de hecho, la forma en que la transformación y el ruteo de mensajes se implementan en WebSphere Message Broker, y son creados utilizando una herramienta gráfica de desarrollo que es parte del propio producto. WebSphere

Message Broker provee varios nodos pre-definidos para construir message-flows: el nodo *MQInput,* que representa una cola de mensajes MQ en la que entran requerimientos al sistema; el nodo *Compute*, que permite operar sobre los campos del mensaje de entrada utilizando extended SQL (ESQL); el nodo *Filter,* que provee lógica *if-then-else;* el nodo *Database,* que permite acceder a bases de datos relacionales via ODBC; y el nodo *MQOutput*, que representa una cola de mensajes de *output*. Existen muchos nodos predefinidos más y, por otra parte, los usuarios pueden desarrollar nodos propios, por ejemplo en Java. El siguiente diagrama esquematiza estos conceptos:



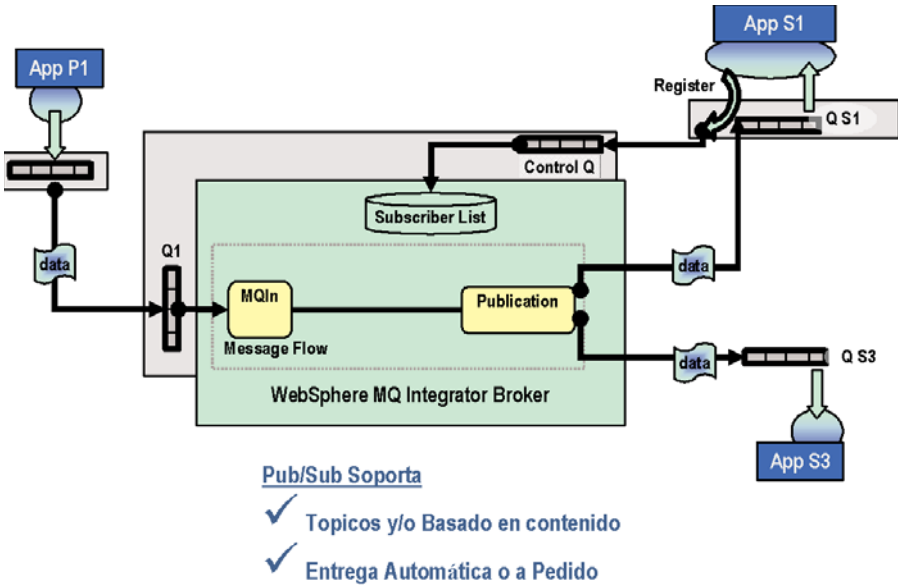
La figura muestra un message-flow que recibe un requerimiento de la Aplicación A en la forma de un mensaje delimitado por comas (nodo MQIn). El message-flow transforma el formato de dicho mensaje a XML en el nodo Compute, almacena el mensaje en una base de datos para auditoría o para un análisis posterior (nodo Database), y finalmente envía el requerimiento a las aplicaciones B y C que serán las encargadas de procesarlo.

Los message-flows pueden ser invocados en la forma de Web Services estándar y, del mismo modo, WebSphere Message Broker puede invocar otros servicios utilizando Web Services.

Hasta acá nos hemos referido implícitamente al intercambio de mensajes entre servicios, utilizando lo que se denomina el modelo push / pull (una aplicación envía un mensaje a otra u otras a través del ESB). Pero, como ya hemos comentado, WebSphere Message Broker soporta ade-

más el modelo basado en eventos, a través de su facilidad de *Publish / Subscribe*. Cuando se utiliza esta facilidad, determinados eventos son publicados en colas de mensajes. Las aplicaciones clientes pueden subscribirse a las novedades que les resulten relevantes, en base al tipo o al contenido de los mensajes publicados. De este modo, un evento que ocurre en determinada aplicación puede ser propagado por el mecanismo de publicación y suscripción a todas las restantes aplicaciones afectadas por dicho evento.

La figura siguiente esquematiza la facilidad de Publish / Subscribe que acabamos de describir:



WebSphere ESB

WebSphere ESB es un nuevo producto que provee un ESB basado en los estándares de Web Services con las siguientes características:

- *Soporte de una variedad de protocolos de transporte*: que si bien no es tan amplia como la soportada por WebSphere Message Broker, incluye HTTP, JMS y WebSphere MQ. Soporta además *protocol switching* entre estos.
- *Utilización de varios modelos de interacción:* tanto el modelo push-pull como el de requerimiento/respuesta y el de publicación/suscripción están soportados.

- *Transformación de formatos:* WebSphere ESB soporta la transformación de formatos basados en Web Services y XML.
- *Acceso a bases de datos relacionales:* tanto para almacenamiento de los mensajes *(logging)* como para enriquecimiento del contenido de los mensajes con información proveniente de bases de datos.

El *WebSphere Integration Developer* (ya mencionado en relación con el ensamblado de procesos BPEL) provee un ambiente gráfico de desarrollo para WebSphere ESB que permite definir la lógica de integración prácticamente sin requerir conocimientos de programación.

WebSphere ESB corre bajo *WebSphere Application Server* y hereda de este las características de escalabilidad, clustering y seguridad. Para administrar la seguridad el producto incluye los componentes *IBM Tivoli Access Manager for e-business* y *Tivoli Directory Server* para autenticación y autorización de usuarios.

WebSphere ESB soporta también conectividad JMS para aplicaciones no-Java a través de clientes específicos que el producto provee para: .Net, C++. [5]

WebSphere MQ

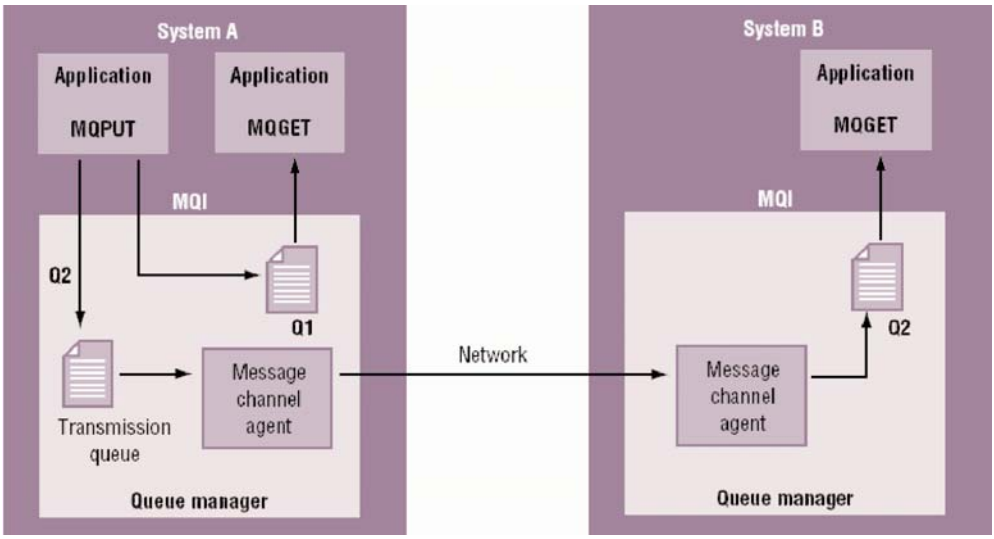
Los servicios en una SOA pueden interactuar con el ESB utilizando Web Services, que tienen la ventaja de ser estándares ampliamente aceptados, lo cual los torna muy ventajosos en muchos casos (por ejemplo: la integración con servicios de otras empresas). Sin embargo, para la interacción entre los servicios internos de una organización, muchas veces es conveniente la utilización de un *middleware* de mensajería que brinde mayor confiabilidad y más control sobre el flujo de mensajes entre las diversas aplicaciones. El *middleware* de mensajería de IBM es el *WebSphere MQ* (anteriormente conocido como *MQSeries*).

WebSphere MQ provee servicios de mensajería entre aplicaciones, conectando virtualmente todo tipo de plataformas a través de su mecanismo de colas de mensajes. Los beneficios de *WebSphere MQ* incluyen:

- Una interface de programación sencilla y consistente. *WebSphere MQ* soporta la misma interface de programación en las más de 35 plataformas soportadas.

- Soporte de JMS (Java Messaging Services), lo que permite a los programas J2EE utilizar mensajería MQ sin abandonar el estándar J2EE.
- Entrega garantizada de los mensajes, una y sólo una vez, con control transaccional.
- Coordinación de transacciones entre recursos MQ (colas de mensajes y canales de conexión) y bases de datos relacionales.
- Gran control para las aplicaciones distribuidas, gracias a su funcionalidad de "aviso de retorno", mediante la cual la aplicación que envía un mensaje puede enterarse cuando este llega a la cola o a la aplicación de destino.
- Reducción de la complejidad de las comunicaciones entre aplicaciones distribuidas, ya que maneja los errores en la red y los reintentos de transmisión.
- Soporte de comunicaciones asincrónicas y sincrónicas entre servicios.

El principal valor de *WebSphere MQ* es brindar un marco confiable para la integración de servicios, aun cuando estos corran sobre plataformas dispares. El siguiente diagrama muestra esquemáticamente el funcionamiento de *WebSphere MQ*:



Como puede verse en el diagrama, MQ permite definir esencialmente dos tipos de objetos: colas de mensajes y canales (conexiones) sobre una red. Las aplicaciones pueden hacer uso de la API de MQ, llamada Message Queue Interface (MQI), para poner mensajes en una cola o para

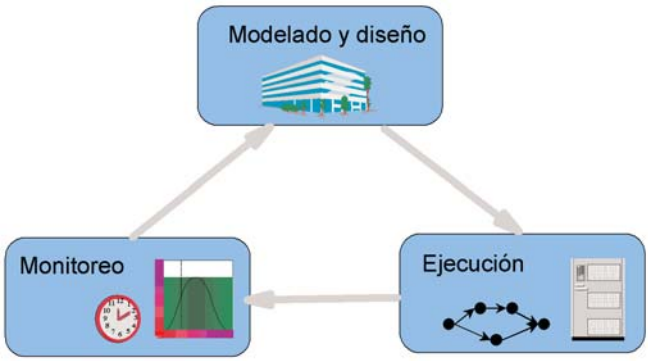
leer mensajes de una cola. Tal como muestra el diagrama, la aplicación del Sistema A está poniendo mensajes en una cola local (Q1) y también en una cola remota (Q2). En este último caso el mensaje es derivado por el MQ automáticamente hacia una cola de transmisión, de donde es tomado por un componente de MQ llamado Message Channel Agent, que se encarga de transmitirlo en forma íntegra a través de la red y de depositar el mensaje en la cola Q2, residente en el Sistema B. Esto brinda un modo muy simple y confiable para que, por ejemplo, el sistema A invoque servicios del sistema B.

MQ soporta mensajes persistentes o no-persistentes. Los primeros son registrados en archivos en disco (logs) de modo que, aunque se produzca una caída del sistema, los mensajes no se pierden. Se proveen también mecanismos de respaldo y recuperación similares a los que proveen las bases de datos para la recuperación de las colas de mensajes ante eventos, tales como la rotura de un disco. Los mensajes no persistentes por su parte utilizan menos recursos, ya que no se registran en disco y brindan por lo tanto una mejor *performance.*

MQ incluye un mecanismo de seguridad fuerte, basado en el protocolo estándar SSL (Secure Sockets Layer), que permite que dos sistemas distribuidos comunicados vía MQ se autentiquen mutuamente antes de comenzar a enviarse mensajes, y que la información contenida en estos viaje encriptada por la red, de modo de asegurar totalmente la confidencialidad. Un nivel aún más alto de seguridad lo brinda un producto complementario, el *Tivoli Access Manager for Business Integration*, que encripta los mensajes que residen en las colas. En este caso, el mensaje es desencriptado recién cuando un programa autorizado lo lee de la cola. [6]

Los servicios de coreografía de procesos

La *Coreografía de Procesos* [7] es la tarea de definir la secuencia y el flujo de información entre componentes de servicios para así formar aplicaciones compuestas que representan procesos de negocio. Una SOA debe proveer un servidor de procesos que permita ejecutar las coreografías de procesos y también las herramientas que permitan diseñar los flujos de proceso y monitorear su ejecución. Como ya hemos visto antes, el modelado de los procesos, su ejecución y su administración o monitoreo forman un ciclo de mejora continua como muestra el diagrama:



La coreografía de procesos incluye dos tipos diferentes de flujos de proceso:

- Microflujos, que son procesos normalmente breves que no incluyen ninguna interacción humana. En estos procesos puede existir la necesidad de volver atrás ante una falla, para lo cual el software debe soportar el concepto de compensación.
- Macroflujos, que son procesos de larga duración (desde varias horas hasta meses), que incluyen interacción humana, o sea, tareas en las que una persona debe tomar una acción. La característica distintiva de estos procesos es que deben persistir en el tiempo, lo que implica que cada cambio de estado debe ser salvado en una base de datos.

En cuanto al lenguaje que se utiliza para especificar los procesos de negocio, una tendencia importante en la actualidad es utilizar el lenguaje estándar *BPEL (Business Process Execution Language)* de amplia aceptación en la industria. Este lenguaje, originalmente propuesto en forma conjunta por IBM, BEA y Microsoft, fue adoptado en 2003 como estándar por OASIS.

El servidor de procesos en la Arquitectura SOA de IBM es el *WebSphere Process Server*. Este es un motor de procesos basado en BPEL, que corre bajo el *WebSphere Application Server* y aprovecha para integrar los servicios a los procesos la funcionalidad del *WebSphere ESB* ya mencionado (el Process Server incluye este componente). Las características más saltantes de WebSphere Process Server son:

- Conectividad estándar basada en WebSphere ESB.
- Modelo de programación diseñado para conectar y utilizar los recursos IT como componentes de servicios.
- Facilidades para reutilización de los componentes de servicios existentes.
- Soporte para procesos dinámicos a través de constructos, tales como: reglas de negocio, máquinas de estado y manejo de eventos.
- Ambiente de ejecución tanto para micro como macro flujos con alta performance y calidad de servicio (tolerancia a fallas, detección de errores, etcétera).
- Herramientas intuitivas de diseño y construcción de procesos, que permiten definir visualmente la secuencia y el flujo de los procesos BPEL con mínimos conocimientos de programación.
- Soporte de compensación para proveer soporte de *"rollback"* a procesos de negocios que requieran esta funcionalidad.
- Posibilidad de incluír código J2EE como parte de los procesos.

El soporte para macro flujos extiende el alcance de BPEL para incluir actividades que requieran interacción humana dentro del proceso de negocios. Los procesos de negocios que requieren intervención humana son procesos interrumpibles y persistentes en el tiempo, ya que una persona puede tardar bastante tiempo en ejecutar una tarea y el proceso debe continuar cuando la persona termine la tarea. El soporte para el flujo de trabajo humano incluye:

- Nodos específicos para representar un paso en un proceso de negocio que se realiza manualmente
- Facilidades para la asignación genérica (basada en roles) de las actividades a las personas.
- Interface gráfica basada en browser para acceder a la lista de items de trabajo *(worklist)* y ejecutar las tareas asignadas o transferirlas a otros usuarios
- Administración de ítem de trabajos para controlar la creación, transferencia y eliminación de ítem de trabajos.

La integración de aplicaciones y datos legacy: WebSphere Adapters

Los adaptadores son componentes de software que permiten una rápida integración de las aplicaciones y tecnologías existentes a una SOA. La mayor ventaja de los adaptadores es que evitan la necesidad de modificar las aplicaciones existentes para poder integrarlas. A través de los adaptadores, pueden integrarse de ese modo a una SOA, aplicaciones legacy, paquetes, tales como SAP, JDE, Siebel, entre otros, y diferentes tecnologías, tales como: bases de datos, e-mail, XML, archivos planos, etcétera. [8]

Dentro de la familia de adaptadores provistos por IBM, llamados *WebSphere Adapters*, existen adaptadores de dos tipos:

- Adaptadores basados en JCA *(J2EE Connector Architecture)*
  - Flat Files
  - JDBC
  - PeopleSoft Enterprise
  - Siebel Business Applications
  - SAP Applications
  - CICS ECI
  - IMS
- Adaptadores basados en JMS *(Java Messaging Services)*

  Estos adaptadores forman un grupo muy extenso que puede verse en el cuadro siguiente:

Application Adapters	Technology Adapters	Mainframe Adapters
• Ariba Buyer • Clarify CRM • eMatrix • i2 • i2 Active Data Warehouse • IndusConnect Framework • Maximo MEA • MetaSolv Applications • mySAP.com • NightFire Applications • Oracle Applications • PeopleSoft • Portal Infranet • QAD MFG/PRO • Retek • Siebel eBusiness Applications • Spirent Applications • Telcordia Applications • WebSphere Commerce	• Adapter for e-mail • FIX Protocol • JMS • JText • JDBC • MQ • MQ Integrator • MQ Workflow • SWIFT • XML • Data Handler for XML • Data Handler for EDI • Web Services	• ADABAS • CICS • DB2 Databases • IMS Transaction Manager • IMS Database Manager • VSAM • Natural • IDMS Database

Los servicios de interacción: los Portales

La idea central de SOA consiste en el concepto de servicios reutilizables que se pueden recombinar con facilidad para crear procesos de negocio. Es natural, por lo tanto, que el *front-end* ideal para SOA consista en servicios de presentación reutilizables, que se puedan recombinar con facilidad para dar lugar a diferentes experiencias de usuario personalizadas e integradas, que representen los intereses y necesidades de cada usuario particular. Una interface de usuario de estas características es lo que se denomina un *portal,* y sus beneficios son una mayor satisfacción de los usuarios de los sistemas y una mayor eficiencia en el acceso a la información.

WebSphere Portal Server es la solución de portal de IBM, basado en tecnología J2EE. Corre sobre *Websphere Application Server*, lo que le otorga una gran flexibilidad y la capacidad de operar sobre múltiples plataformas. [9]

WebSphere Portal Server provee su propio ambiente de administración incluyendo un conjunto de herramientas de desarrollo, administración y personalización que permiten crear y gestionar portales seguros y flexibles. Estas herramientas aplican a la gestión de contenidos web, la configuración de *portlets*, la administración de usuarios y permisos de acceso al portal, entre otras prestaciones.

Vamos a describir a continuación algunos de los aspectos más relevantes de WebSphere Portal Server:

- Portlets

  Los *portlets* son el corazón de WebSphere Portal Server. Se trata de aplicaciones reutilizables, similares a los *servlets*, que se visualizan en el portal como porciones específicas de una página *web*. Existe un catálogo de *portlets* disponibles para WebSphere Portal Server, que incluye una gran cantidad de ellos, provistos no solo por IBM sino también por otras empresas, tales como SAP, Siebel, Citrix, entre otras.

- Facilidades de integración

  Los *portlets* incluyen la funcionalidad de *Click to Action*. Esta característica se refiere a la capacidad de transmitir la información de un *portlet* a otro, en tiempo real. Esta facilidad es la que permite a los usuarios acceder a información integrada aunque los recursos que utilicen se encuentren materialmente distribuidos en diferentes aplicaciones y bases de datos.

- Seguridad

  WebSphere Portal Server incluye facilidades para definir los usuarios y grupos de usuarios, y para administrar sus permisos de acceso. El portal provee también soporte para *single sign-on*.

- Personalización

  WebSphere Portal Server provee facilidades tanto para que los administradores como los usuarios finales puedan personalizar el contenido y el diseño de las páginas del portal.

- Administración de contenidos

  IBM provee una herramienta de administración de contenidos llamada *Web Content Manager*. Esta herramienta permite que los usuarios de negocios, sin ningún conocimiento específico de programación, puedan crear sitios web completos en base a plantillas y componentes reutilizables. La herramienta provee también funciones colaborativas para facilitar el trabajo de diseño en equipo y *workflows* para los procesos de aprobación de contenidos. Web Content Manager permite además conservar versiones pasadas del sitio y la historia de las aprobaciones.

- Facilidades colaborativas

  WebSphere Portal Server incluye facilidades colaborativas avanzadas, tales como mensajería instantánea, *teamrooms* virtuales y reuniones electrónicas *(e-meetings).*

Conclusión

Hemos planteado los conceptos generales de la arquitectura de integración basada en servicios de IBM, basada en las ideas de la arquitectura orientada a servicios (SOA). Solo una arquitectura como esta puede brindar el grado de integración que las empresas necesitan hoy y la flexibilidad necesaria para poder rediseñar los procesos rápidamente, de modo que puedan adaptarse con facilidad a nuevos requerimientos.

La SOA resuelve el problema de la integración transversal atacándolo en la raíz: reemplaza el acoplamiento fuerte de las aplicaciones por un conjunto de servicios independientes vinculados por un ESB. El ESB provee acoplamiento débil, de manera que los servicios actúen como verdaderas cajas negras. Si, por ejemplo, la implementación interna de un servicio llegase a cambiar, modificando algunas reglas de negocios a nivel del ESB puede lograrse que todo siga funcionando en forma transparente para los restantes servicios que no se verían afectadas en absoluto.

En definitiva, de lo que se trata es de encapsular la implementación interna de cada servicio, lo cual facilita enormemente su reutilización. Si se le agrega a esto la facilidad de orquestar procesos complejos a partir de los servicios individuales, se puede entender por qué esta arquitectura puede ofrecer la flexibilidad requerida.

La integración de las funciones y los procesos de negocios se complementan con la integración de la presentación asociada con el concepto de portal. En efecto, los portales brindan a sus usuarios una experiencia integrada de los sistemas de la empresa y facilitan la interacción de los usuarios con ellos, mejorando así su productividad.

¿Cómo comenzar con una SOA? Una manera de hacerlo es comenzar con la integración de los procesos. Se deben identificar algunos procesos para un primer proyecto piloto, relevando todas sus características (flujo, información requerida en cada paso, aplicaciones participantes, etcétera).

Un paso importante es la identificación de servicios en las aplicaciones existentes. Esto puede ser relativamente sencillo o bastante complejo, dependiendo del diseño de dichas aplicaciones. Como se comentó con anterioridad, hay dos enfoques posibles para esto que consisten en:

- Modificar el código de la aplicación para permitir que sus funciones sean invocadas a través del mecanismo implementado por el ESB.
- Implementar los servicios aplicativos en un adaptador, que a su vez invocará a las funciones correspondientes a través de las interfaces nativas de la aplicación.

Cuando se adopta el primer enfoque, muchas veces las herramientas de desarrollo brindan facilidades para convertir las funciones aplicativas en servicios. Esto es particularmente cierto en el caso de las aplicaciones J2EE, pero también lo es en muchos otros casos (por ejemplo, en transacciones CICS).

El segundo enfoque es valioso cuando resulta muy costoso o riesgoso modificar el código o cuando simplemente no se dispone de este (como ocurre en el caso de los paquetes aplicativos). Como hemos visto, IBM provee múltiples adaptadores para varios paquetes y tecnologías de uso frecuente.

El ESB es el corazón mismo de una SOA y es una de las primeras partes que debe implementarse. Evidentemente, si se comienza con algunos procesos piloto, debe considerarse también la implementación del

servidor de procesos y de las herramientas de modelado. Elementos adicionales tales como las herramientas de monitoreo pueden tal vez ser implementadas a posteriori.

La experiencia general muestra que este tipo de soluciones de integración tienden a crecer rápidamente, de modo que una buena práctica es comenzar con una implementación simple y preparar el ambiente operativo para dicho crecimiento.

IBM tiene una gran experiencia en la implementación de este tipo de arquitecturas y de soluciones de integración en general. Además dispone de un amplio y variado portafolio de productos de software, específicamente diseñado para facilitar la implementación de arquitecturas orientadas a servicios, algunos de cuyos componentes hemos descrito a lo largo del presente documento. IBM dispone también de una amplia gama de servicios de consultoría e implementación, que se brindan tanto con personal propio como con asociados de negocio, para acompañar a todos los clientes que deseen comenzar con la implantación de soluciones de integración.

Notas

[1] Artículo "What is SOA?" [en línea], en IBM DeveloperWorks. <http://www-128.ibm.com/developerworks/webservices/newto>.

[2] Hay muchísima bibliografía sobre Web Services. Para una excelente introducción ver "What is Web Services?" [en línea], en IBM DeveloperWorks

<http://www-128.ibm.com/developerworks/webservices/newto/websvc.html>

Para una exposición más detallada se pueden consultar:

Van de Putte, Geert et al. "Using Web Services for Business Integration" (IBM ITSO), en especial los capítulos 1 y 2.

Endrei, Mark et al. "SOA and Web Services" (IBM ITSO).

Erl, Thomas. "Service Oriented Architecture" (Prentice-Hall).

[3] El Enterprise Service Bus es, rigurosamente hablando, un patrón de diseño de soluciones SOA. Para ver en detalle sus características puede consultarse de Keen, Martin et al. "SOA with an Enterprise Service Bus" (IBM ITSO).

[4] Los temas tratados en esta sección pueden ampliarse consultando a Wahli, Ueli et al. "Business Process Management, Modeling through Monitoring" (IBM ITSO).

[5] Sadtler, Carla et al. "Enabling SOA using WebSphere messaging" (IBM ITSO). Este libro contiene capítulos específicos para WebSphere ESB y para WebSphere Message Broker.

[6] El mejor libro para estudiar la funcionalidad de WebSphere MQ sigue siendo el de Blakeley, Harris y Lewis, *Messaging and Queuing Using the MQ Interface* (McGraw-Hill). Otro texto excelente y más actualizado es el de Saida Davies y Peter Broadhurst: "WebSphere MQ V6 Fundamentals", (IBM ITSO).

[7] Keen, Martin et al., "Patterns: Building Serial and Parallel Processes for WebSphere Process Server" (IBM ITSO).

[8] Información adicional sobre los WebSphere Adapters puede obtenerse en <http://www-306.ibm.com/software/integration/wbiadapters/library/>.

[9] Chandran, Anup et al. "Architecting Portal Solutions" (IBM ITSO).

Nota: Todos los libros publicados por IBM ITSO (International Technical Support Organization) se pueden leer en línea o descargarse en formato pdf desde la siguiente dirección web: <http://www.ibm.com/redbooks.